Numerical Simulation guided Lazy Abstraction Refinement for Nonlinear Hybrid Automata

Sumit Kumar Jha[1]

Computer Science Department, Carnegie Mellon University
5000 Forbes Avenue, Pittsburgh, PA15213, USA
jha+@cs.cmu.edu

Abstract. This draft suggests a new counterexample guided abstraction refinement (CEGAR) framework that uses the combination of numerical simulation for nonlinear differential equations with linear programming for linear hybrid automata (LHA) to perform reachability analysis on nonlinear hybrid automata. A notion of $\epsilon-$ structural robustness is also introduced which allows the algorithm to validate counterexamples using numerical simulations.

1 Introduction

The model checking of hybrid automata remains a challenge and the existing tools [6,4] do not scale up to the needs of the industry. Because of the well known fundamental undecidability results [5], the model checking of general hybrid automata often proceeds by building successive tighter approximations to these hybrid automata in a relatively easy-to-analyze fragment of hybrid automata like Linear Hybrid Automata [7]. Theoretical results about the asymptotic completeness of this approximation procedure form the backbone of such a strategy behind the model checking of nonlinear hybrid automata.

There has been considerable interest in applying Counterexample Guided Abstraction Refinement (CEGAR), which works so well with discrete systems, to the problem of hybrid system verification [2]. There has also been some exploration of using fragments instead of counterexamples during abstraction refinement [3] and the application of CEGAR specifically to LHA [8]. However, our ongoing work makes the following new contributions to the abstraction refinement based analysis of hybrid systems.

– We address the problem of abstraction refinement for nonlinear hybrid automata and use CEGAR to construct successively refined LHA approximations. Our refinement is lazy and hence, refines some parts of the state space more finely than others.
– We use the distance between a feasible path in the abstract linear hybrid automata and the numerically simulated trajectory in the nonlinear hybrid automata to refine those locations in the LHA that do not faithfully represent the behavior of the nonlinear hybrid automata.

– We define a structural notion of robustness and use it to present a counterexample validation algorithm (for a rich class of nonlinear hybrid automata) using linear programming [9]. Hence, it is possible to detect reachability of a bad state even before the abstraction refinement loop terminates.

2 Background on LP based path feasibility analysis of LHA

Informally, a linear hybrid automaton is a conventional automaton extended with a set of continuous variables. The states of the automaton called *locations* are annotated with a change rate for each continuous variable such as $\dot{x} = [a, b]$ (x is a variable, and $[a, b]$ is a rational interval), and the transitions of the automaton are labeled with constraints on the variables such as $a \leq \sum_{i=0}^{m} c_i x_i \leq b$ and /or with reset actions such as $x := c$ (x_i and x are variables, a, b, and c_i are real numbers). Such linear hybrid automata are essentially equivalent to the definition given in [5]. It is known that this subclass of linear hybrid automata are sufficiently expressive to allow asymptotic completeness of the abstraction process for a general hybrid automata. " A restricted form of linear phase portrait approximations are asymptotically complete, namely, when all automaton constraints are over-approximated using independent, rational lower and upper bounds on the values and derivatives of each variable" [1].For simplicity, we suppose that in any linear hybrid automaton considered in this paper, there is just one initial location with no initial conditions and no transitions to the initial location (we assume that each variable with an initial value is reset to the initial value by the transitions from the initial location).

Definition 1. A linear hybrid automaton is a tuple $H = (X, V, E, v_I, \alpha, \beta)$, where

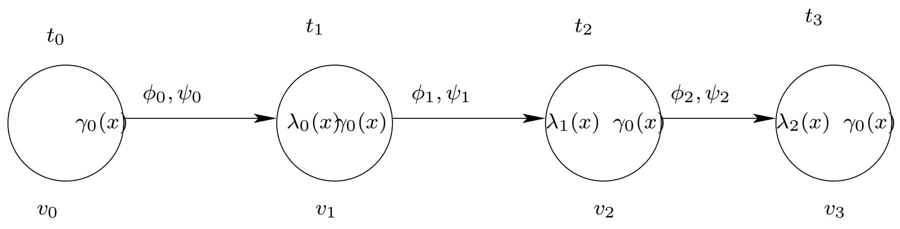
– X is a finite set of real-valued variables.
– V is a finite set of *locations*.
– E is *transition* relation whose elements are of the form (v, ϕ, ψ, v') where v, v' are in V, ϕ is a set of *guards or variable constraints* of the form $a \leq \sum_{i=0}^{m} c_i x_i \leq b$, and ψ is a set of *reset actions* of the form $x := c$ where $x_i \in X$ $(0 \leq i \leq m)$, $x \in X$, a, b and c_i $(0 \leq i \leq m)$ are real numbers, and a and b may be ∞.
– v_I is an *initial* location.
– α is a labeling function which maps each location in $V - \{v_I\}$ to a *state invariant* which is a set of variable constraints of the form $a \leq \sum_{i=0}^{m} c_i x_i \leq b$ where $x_i \in X$ $(0 \leq i \leq m)$, $y \in X$, a, b, and c_i $(0 \leq i \leq m)$ are real numbers, a and b may be ∞.
– β is a labeling function which maps each location in $V - \{v_I\}$ to a set of *change rates* which are of the form $\dot{x} = [a, b]$ where $x \in X$, and a, b are rational numbers $(a \leq b)$. For any location v, for any $x \in X$, there is one and only one change rate definition $\dot{x} = [a, b] \in \beta(v)$.

□

For a linear hybrid automaton $H = (X, V, E, v_I, \alpha, \beta)$, a *path segment* is a sequence of locations

$$v_1 \xrightarrow{(\phi_1,\psi_1)} v_2 \xrightarrow{(\phi_2,\psi_2)} \ldots \xrightarrow{(\phi_{n-1},\psi_{n-1})} v_n$$

which satisfies $(v_i, \phi_i, \psi_i, v_{i+1}) \in E$ for each i $(1 \leq i \leq n-1)$. A *path* in H is a path segment starting at v_I. The behavior of linear hybrid automata can be represented by *timed sequences*. Any timed sequence is of the form $(v_1, t_1)\hat{}(v_2, t_2)\hat{}\ldots\hat{}(v_n, t_n)$, where v_i $(1 \leq i \leq n)$ is a location and t_i $(1 \leq i \leq n)$ is a nonnegative real number, which represents a behavior of an automaton that the system starts at the initial location and changes to the location v_1, stays there for t_1 time units, then changes to the location v_2 and stays in v_2 for t_2 time units, and so on.



Definition 2. *[9]* For a linear hybrid automaton $H = (X, V, E, v_I, \alpha, \beta)$, a timed sequence $(v_1, t_1)\hat{}(v_2, t_2)\hat{}\ldots\hat{}(v_n, t_n)$ represents a behavior of H if the following condition is satisfied:

– there is a path in H of the form

$$v_0 \xrightarrow{(\phi_0,\psi_0)} v_1 \xrightarrow{(\phi_1,\psi_1)} \ldots \xrightarrow{(\phi_{n-1},\psi_{n-1})} v_n \,;$$

– $t_1, t_2, \ldots, t_n$ satisfy all the variable constraints in ϕ_i $(1 \leq i \leq n-1)$, i.e. for each variable constraint $a \leq c_0x_0 + c_1x_1 + \ldots + c_mx_m \leq b$ in ϕ_i,

$$\delta_k \leq \gamma_i(x_k) \leq \delta'_k \text{ for any } k \ (0 \leq k \leq m), \text{ and}$$
$$a \leq c_0\gamma_i(x_0) + c_1\gamma_i(x_1) + \ldots + c_m\gamma_i(x_m) \leq b$$

where $\gamma_i(x_k)$ $(0 \leq k \leq m)$ represents the value of the variable x_k when the automaton stay at v_i with the delay t_i; and, similarly,

– $t_1, t_2, \ldots, t_m, \gamma_i(x_k), \lambda_i(x_k)$ satisfy the state invariant for each location v_i $(1 \leq i \leq n)$, where $\gamma_i(x_k)$ $(0 \leq k \leq m)$ represents the value of the variable x_k when the automaton stay at v_i with the delay t_i, and $\lambda_i(x_k)$ $(0 \leq k \leq m)$ represents the value of the variable x_k after leaving state v_i and after the reset conditions have been applied.

Now, we use linear programming to test the feasibility of a single path for the reachability analysis of linear hybrid automata. Let $H = (X, V, E, v_I, \alpha, \beta)$ be a linear hybrid automaton, , and ρ be a path in H of the form

$$v_0 \xrightarrow{(\phi_0,\psi_0)} v_1 \xrightarrow{(\phi_1,\psi_1)} \dots \xrightarrow{(\phi_{n-1},\psi_{n-1})} v_n$$

where $v_n = v$. For any timed sequence of the form $(v_1, t_1)\hat{}(v_2, t_2)\hat{} \dots \hat{}(v_n, t_n)$, if ρ is feasible, then the following condition must hold:

- $t_1, t_2, \dots, t_n$ satisfy all the variable constraints in $\phi_i (0 \le i \le n)$, and
- $t_1, t_2, \dots, t_n$ satisfy all the variable constraints in $\alpha(v_i)$ $(1 \le i \le n)$,

which form a group of linear inequalities on $t_1, t_2, \dots, t_n, \gamma_i(x_k), \lambda_i(x_k)$ (see Definition 2), denoted by $\Theta(\rho)$ or $LP_\rho(t_i, \gamma_i(x_k), \lambda_i(x_k))$. It follows that we can check if ρ is a feasible path by checking if the group $\Theta(\rho)$ (or $LP_\rho(t_i, \gamma_i(x_k), \lambda_i(x_k))$) of linear inequalities has a solution, which can be solved by linear programming [9].

3 Background on Abstraction of Affine dynamics by LHA

Given a general hybrid system $H = (X, V, E, v_I, \alpha, \beta)$ where X is a finite set of real-valued variables, V is a finite set of *locations*, E is *transition* relation whose elements are of the form (v, ϕ, ψ, v') where v, v' are in V, ϕ is a set of *guards or variable constraints* of the form $a \le \sum_{i=0}^{m} c_i x_i \le b$, and ψ is a set of *reset actions* of the form $x := c$ where $x_i \in X$ $(0 \le i \le m)$, $x \in X$, a, b and c_i $(0 \le i \le m)$ are real numbers, and a and b may be ∞, v_I is an *initial* location, α is a labeling function which maps each location in $V - \{v_I\}$ to a *state invariant* which is a set of variable constraints of the form $a \le \sum_{i=0}^{m} c_i x_i \le b$ where $x_i \in X$ $(0 \le i \le m)$, $y \in X$, a, b, and c_i $(0 \le i \le m)$ are real numbers, a and b may be ∞, β is a labeling function which maps each location in $V - \{v_I\}$ to a set of *change rates* which are of the form $\dot{x_i} = f(x_0, x_1, \dots, x_m, \dot{x}_0, \dot{x}_1, \dots, \dot{x}_m, a_0, a_1 \dots a_m)$ where $x_0, x_1 \dots x_m \in X$, and $a_0, a_1 \dots, a_n$ are real numbers). For any location v, for any $x \in X$, there is one and only one change rate definition.

We construct (in a fashion similar to [6,4]) a linear hybrid automata H_a which is an over-approximate abstraction of the general hybrid automata H. We define an operator *split* which divides a location into smaller locations and use this to divide each location v_i of the original hybrid automaton.

Definition 3. *Let v be a location and $x_i \in X$ be a variable in a hybrid automata $H = (X, V, E, v_i, \alpha, \beta)$. Suppose $In_v = \{v_0^{in}, v_1^{in} \dots v_n^{in}\}$ be the locations from which there is a transition into the location v and $Out_v = \{v_0^{out}, v_1^{out} \dots v_m^{out}\}$ be the locations to which there is a transition from v. Also, C be a set of linear constraints on X. Then, the operator split (H, v, C) constructs a new hybrid automata $H' = (X, V', E', v_i, \alpha', \beta')$, where*

- $V' = V \cup \{v', v''\} \setminus v$

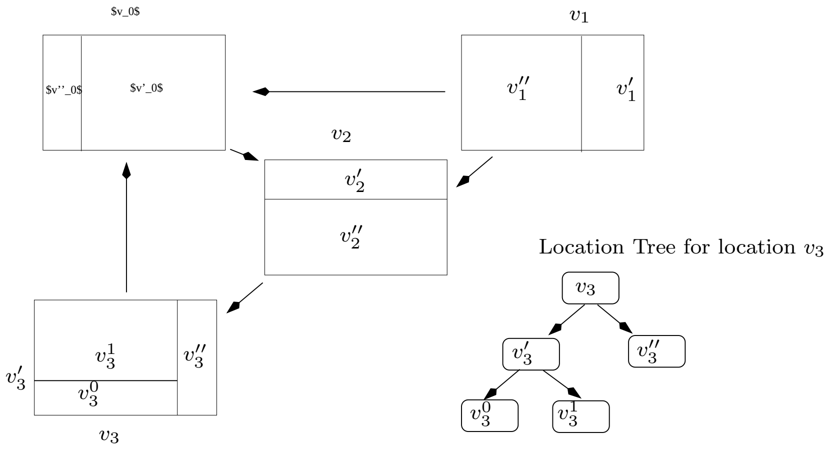
– $\alpha' = \alpha \cup \{\alpha(v'), \alpha(v'')\} \setminus \{\alpha(v)\}$, *where*

- $\alpha(v') = \alpha(v) \cup \{C\}$
- $\alpha(v'') = \alpha(v) \cup \{\neg C\}$

– $E' = E \setminus (\{(v_{in}, \phi_{v_{in},v}, \psi_{v_{in},v}, v) | v_{in} \in In_v\} \cup \{(v, \phi_{v,v_{out}}, \psi_{v,v_{out}}, v_{out}) | v_{out} \in Out_v\}) \cup (\{(v_{in}, \phi_{v_{in},v}, \psi_{v_{in},v}, v') | v_{in} \in In_v\} \cup \{v', \phi_{v,v_{out}}, \psi_{v,v_{out}}, v_{out} | v_{out} \in Out_v\} \cup \{(v_{in}, \phi_{v_{in},v}, \psi_{v_{in},v}, v'') | v_{in} \in In_v\} \cup \{v'', \phi_{v,v_{out}}, \psi_{v,v_{out}}, v_{out} | v_{out} \in Out_v\}) \cup \{(v', \alpha(v', v'') = C, \{\}, v'')\ , (v'', \alpha(v'', v') = \neg C, \{\}, v')\}$

– $\beta' = \beta \cup \{\beta(v'), \beta(v'')\} \setminus \beta(v)$, *where*

- $\beta(v') = \beta(v)$
- $\beta(v'') = \beta(v)$



We call the new locations v' and v'' as the children of v. In particular, $v' = child(v, C)$ and $v'' = child(v, \neg C)$, where C is the set of linear constraints used to split v. We also call v as the parent of v' and v''. Thus, the *split* operator naturally defines a tree of locations that we call the *location tree* , where the children location are formed by splitting the parent location.

Definition 4. *LHA-approximation to a general hybrid automata: Given a general hybrid automata $H = (X, V, E, v_i, \alpha, \beta)$, $H_a = (X, V_a, E_a, V_{i_a}, \alpha_a, \beta_a)$ is a LHA-approximation iff*

– *There exists a hybrid automata $H' = (X, V', E', V'_i, \alpha', \beta')$, where $H' = split^n(H)$.*
– $V_a = V', E_a = E', V'_i = V_{i_a}, \alpha' = \alpha_a$
– $\forall v \in V_a, \beta_a(v) \supset \beta'_v$ *and if $c \in \beta_a(v)$, then c is of the form $x := [a, b]$, where $a, b \in \mathcal{R}$.*

4 Definitions

The linear hybrid automaton H_a is an over-approximate approximation of the general hybrid automata H. A path $\rho = \{v_0, v_1, \dots v_m\}$, where $v_i \in V$ is said to *exist* in H_a if $(v_i, v_{i+1}) \in E, 0 \leq i < m$.

Consider a path $\rho = \{v_0, v_1, \dots v_m\}$ that *exists* in the abstract linear hybrid automata model H_a and let $LP_\rho(t_i, \gamma_i(x_k), \lambda_i(x_k))$ be the linear program corresponding to the path. If the linear program LP_ρ has a feasible solution, then the path is said to be *feasible* in the abstract model i.e. the linear hybrid automaton; otherwise it is said to be *infeasible* .

Definition 5. *Trace: Given a feasible path ρ in the abstract linear hybrid automaton model H_a, the feasible solution to the LP program $LP_\rho(t_i, \gamma_i(x_k), \lambda_i(x_k))$ is called a trace of H_a.*
We write trace$(H_a) =< (v_0, \lambda_0(x_0)\ , \lambda_0(x_1) \dots \lambda_0(x_n), \gamma_0(x_0), \gamma_0(x_1), \dots \gamma_0(x_n), t_0), (v_1, \lambda_1(x_0),\ \lambda_1(x_1)\ \dots\ \lambda_1(x_n),\ \gamma_1(x_0),\ \gamma_1(x_1),\ \dots \gamma_1(x_n), t_1) \dots \dots (v_m, \lambda_m(x_0), \lambda_m(x_1) \dots \lambda_m(x_n),\ \gamma_m(x_0), \gamma_m(x_1),\ \dots \gamma_m(x_n), t_m) >$.

It is known [9] that the trace obtained by the linear program is a real execution trace of the over-approximate linear hybrid automata.

Definition 6. *Concretization of a path: Consider a path $\rho = \{v_0, v_1, \dots v_m\}$ that is feasible in the abstract linear hybrid automata model H_a. Then, the concretization of this path in the original hybrid automata H is the trace $\rho_{concrete} = \{v_0^{root}, v_1^{root} \dots v_m^{root}\}$, where v_i^{root} is the root of the location tree in which v_i is a leaf.*

As the *split* operator forms a tree of locations in the abstract linear hybrid automata, the root of the location tree is known and the *concretization of an abstract path* is well defined.

Definition 7. *Concretization of a trace: Consider a trace $tr =< (v_0, \lambda_0(x_0)\ , \lambda_0(x_1) \dots \lambda_0(x_n), \gamma_0(x_0)\ , \gamma_0(x_1), \dots \gamma_0(x_n), t_0), (v_1,\ \lambda_1(x_0),\ \lambda_1(x_1) \dots \lambda_1(x_n), \gamma_1\ (x_0), \gamma_1(x_1), \dots \gamma_1(x_n), t_1) \dots \dots (\ v_m, \lambda_m(x_0), \lambda_m(x_1) \dots \lambda_m(x_n),\ \gamma_m(x_0), \gamma_m(x_1), \dots \gamma_m(x_n), t_m) >$ corresponding to the path $\rho = \{v_0, v_1, \dots v_m\}$ that is feasible in the abstract linear hybrid automata model H_a. Then, the concretization of this trace in the original hybrid automata H is the trace $tr_{concrete} =< (v_0^{root}, \lambda_0(x_0), \lambda_0(x_1) \dots \lambda_0(x_n), \gamma_0(x_0), \gamma_0(x_1), \dots\ \gamma_0(x_n), t_0),\ (v_1^{root}, \lambda_1(x_0), \lambda_1(x_1)\ \dots \lambda_1(x_n), \gamma_1(x_0), \gamma_1(x_1), \dots\ \gamma_1(x_n), t_1)\ \dots \dots (v_m^{root}, \lambda_m(x_0),\ \lambda_m(x_1)\ \dots \lambda_m(x_n), \gamma_m(x_0), \gamma_m(x_1), \dots\ \gamma_m(x_n), t_m) >$, where v_i^{root} is the root of the location tree of which v_i is a leaf.*

Definition 8. *ϵ-Simulation Trajectory: Given a valuation of X i.e. $Val_0(X) = (x_0 = x0, x_1 = x1, \dots x_n = xn)$ in a location v of a general hybrid system $H = (X, V, E, v_i, \alpha, \beta)$, then $\tau(x0, x1, \dots xn, v, t)$ is said to be an $\epsilon-$ simulation trajectory for location v with initial valuation $Val_0(X)$ iff*

– $\tau(t = 0) = (x0, x1, \dots, xn)$

– *if $f(x_0, x_1, \ldots, x_n, t)$ is the solution to the initial value problem $(\beta(v), Val_0(X))$, then $f(t) - \epsilon \leq \tau(t) \leq f(t) + \epsilon$*

It is known that numerical techniques can solve the initial value problem for ODEs (including non-linear ODEs) quiet efficiently.

Definition 9. *$\epsilon-$ Hybrid Simulation Trajectory: Given an initial valuation of X i.e. $Val_0(X) = (x_0 = x0, x_1 = x1, \ldots x_n = xn)$ and a path $\rho = \{v_0, v_1 \ldots v_m\}$ in a general hybrid system $H = (X, V, E, v_i, \alpha, \beta)$, then $\tau(x_0, x_1, \ldots x_n) = f(t)$ is said to be an $\epsilon-$ hybrid simulation trajectory iff*

– *$\tau(0) = Val_0(X)$ and $Val_0(X) \in \alpha(v_0)$*
– *if $x_k := e \in \psi(v_i, v_{i+1})$ then $\tau(x_k, \sum_0^i(t_i)+ = e)$ else $\tau(x_k, \sum_0^i(t_i)+) = \tau(x_k, \sum_0^i(t_i)-)$*
– *Before executing the jump (v_i, v_{i+1}), $\tau(x_k, \sum_0^i(t_i)-)$ satisfies every precondition in $\phi(v_i, v_{i+1})$*
– *Within each location v_i where the timed path has spent time t_i,*
  - *$\forall t, t_i < t < t_{i+1}, \tau(v, \sum_0^i(t_i) + t)$ is an $\epsilon-$ simulation trajectory for location v_i with initial valuation $Val_0 = \tau(\sum_0^i(t_i))$.*

Definition 10. *Guided Simulation Trajectory of the concretization of a trace : An $\epsilon-$ hybrid simulation trajectory τ is said to be a Guided Simulation Trajectory of a concretized trace $tr_{concrete}$ iff*

– *The initial valuation of X i.e. $Val_0(X) = (x_0 = x0, x_1 = x1, \ldots x_n = xn)$ for the trajectory τ is the initial point in the concretized trace $tr_{concrete}$.*
– *The $\epsilon-$ hybrid simulation trajectory τ corresponds to the path $\rho = \{v_0, v_1 \ldots v_m\}$ corresponding to $tr_{cocnrete}$*

5 CEGAR based Refinement of the abstract linear hybrid automata

The CEGAR algorithm repeatedly constructs LHA over-approximations to the given (possibly nonlinear) hybrid system and then asks a LHA analysis engine if the over-approximate LHA admits any counterexample. If it does not, we are done and we report that the original hybrid system has no counterexample either. Otherwise, we take the reported counterexample of the over-approximate LHA and attempt to validate it using numerical simulation. If we succeed in validating the counterexample, we report an error that the bad state is reachable and STOP. Otherwise, we find a location where we need to split the nonlinear hybrid automata and then rebuild a more precise over-approximate abstraction.

```
Algorithm for CEGAR
(Input: Nonlinear Hybrid Automata A. Output: Error  No error
    1. $A_0 = A$; i := 0; L = Universe.
    2. $LHA_i$ = LHA-approximation ($A_i$)
    3. $\mathcal{L}(LHA_i)$ := Language of $LHA_i$. $\mathcal{L}$ represents the set of potential counterexamples in $A_i$ (finitely expressible as a regular expression) [8].
    4. $L = L \cap \mathcal{L}(LHA_i)$
    5. If $L$ is empty, report "BAD STATES NOT REACHABLE" and stop.
    6. Pick a counterexample $ce$ in $L$.
    7. Validate the counterexample $ce$ in the original hybrid automata $A$.
    8. If $ce$ is validated in A, stop and report that ERROR STATE IS REACHABLE.
    9. Compute a refinement operator $split$, and $A_{i+1} = split(A_i)$. Also, compute $L = split(L)$.
    10. i := i + 1
    11. Loop to Step 2.
```

5.1 Counterexample Validation and Structural Robustness

Let $v_0, \gamma_0(x_0), \gamma_0(x_1) \ldots \gamma_0(x_n), t_0$ be the initial point in a concretized trace $tr_{concrete}$ for the abstraction i.e. the linear hybrid system H_a corresponding to the general hybrid system H. Let $\tau_{(v_0,x_0,x_1,\ldots x_n)}$ be the $\epsilon-$ Hybrid Simulation Trajectory starting from this initial point.

Definition 11. *$\epsilon-$ Structurally Robust Hybrid System: A hybrid system $H = (X, V, E, v_i, \alpha, \beta)$ is said to be structurally robust iff*

- *$\forall i, (v_i, \phi, \psi, v_{i+1}) \in E$, every constraint c in ϕ is satisfied by at least a dense set of size ϵ i.e. If $S = \{Val = (x_0, x_1, \ldots x_n) | Val$ satisfies $c\}$, then $max_{a \in S} min_{b \in S} d(a, b) > \epsilon$.*
  In particular, we allow only sampled comparisons $x :=_\epsilon c$, which is a shorthand for $\lfloor \frac{c}{\epsilon} \rfloor \times \epsilon < x < \lceil \frac{c}{\epsilon} \rceil \times \epsilon$.

Definition 12. *ϵ Robust Hybrid Simulation Trajectory: Given an initial valuation of X i.e. $Val_0(X) = (x_0 = x0, x_1 = x1, \ldots x_n = xn)$ and a path $\rho = \{v_0, v_1 \ldots v_m\}$ in a general hybrid system $H = (X, V, E, v_i, \alpha, \beta)$, then $\tau(x_0, x_1, \ldots x_n) = f(t)$ is said to be an $\epsilon-$ robust hybrid simulation trajectory iff*

- *$\tau(0) = Val_0(X)$ and $Val_0(X) \in \alpha(v_0)$*
- *if $x_k := e \in \psi(v_i, v_{i+1})$ then $\tau(x_k, \sum_0^i (t_i)+ = e)$ else $\tau(x_k, \sum_0^i (t_i)+) = \tau(x_k, \sum_0^i (t_i)-)$*
- ***Before executing the jump** (v_i, v_{i+1})**,** $\tau(x_k, \sum_0^i (t_i)-)$ **satisfies every precondition in** $\phi(v_i, v_{i+1})$ $\epsilon-$ *robustly .*

  - *A linear constraint c is $\epsilon-$ robustly satisfied by $X = (x_0, x_1, \ldots)$ iff for every X' such that $d(X, X') \leq \epsilon)$, $c(X')$ is true.*
- *Within each location v_i where the timed path has spent time t_i,*
  - *$\forall t_i < t < t_{i+1}, \tau(v, \sum_0^i(t_i)+t)$ is an $\epsilon-$ simulation trajectory for location v_i with initial valuation $Val_0 = \tau(\sum_0^i(t_i))$.*

Theorem 1. *If $\tau_{(v_0,x_0,x_1,\ldots x_n)}$ be the $\epsilon-$ Robust Hybrid Simulation Trajectory starting from the initial valuation $Val_0 = \gamma_0(x_0), \gamma_0(x_1) \ldots \gamma_0(x_n)$, and H be a $\epsilon-$ structurally robust hybrid system, then $tr_{concrete}$ corresponds to a* ***real*** *counterexample for the hybrid system H.*

Proof. The proof follows from the definition of $\epsilon-$ robust hybrid automata and the notion of $\epsilon-$ hybrid simulation trajectory.

5.2 Simulation Based Abstraction Refinement

Consider the *concretization of a trace $tr_{concrete}$* with respect to the general hybrid automata H obtained from a *trace tr* of the abstract linear hybrid automata H_a. Also, consider the *guided hybrid simulation trajectory $\tau_{tr_{concrete}}$* corresponding to the concretization of the trace tr with respect to the general hybrid automata H.

Metrics for distance between *trace* and *trajectory* We define two distance metrics between a trace and the corresponding guided hybrid simulation trajectory.

- $D(t) = d(\tau_{tr_{concrete}}(t), tr_{concrete}(t))$.
  This is simply a distance metric between corresponding points on the trace and the trajectory. The metric d may be the Euclidean distance metric or the Manhattan distance metric (linear function).
- $D'(t) = d'(d(\tau_{tr_{concrete}}(t), tr_{concrete}(t)), d(\tau_{tr_{concrete}}(t-), tr_{concrete}(t-))$
  This metric measures how rapidly the guided hybrid simulation trajectory is moving away from the trace. The metric d may be the Euclidean distance metric or the Manhattan distance metric (linear function), while the metric d' may be the real difference. $t-$ represents the last instant of time for which the value of the concretized trace is known.

Strategies to choose the location to be refined Let t_i be the discrete point on the *concretization of a trace* i.e. on $tr_{concrete}$ for which $tr_{concrete}(t)$ is known from the solution of the LP problem. There are few different strategies to choose the location in the approximate linear hybrid automata, where one needs to refine the abstract hybrid automata H_a.

- $min_i|D(t_i)| > \epsilon$, where ϵ is an empirically determined constant.
- $min_i|D'(t_i) - D'(t_{i-1})| > \epsilon$, where ϵ is an empirically determined constant.

– $min_i|D'(t_i)/D'(t_{i-1})| > \epsilon$, where ϵ is an empirically determined constant.

After finding out the point t_i where one needs to refines the location, the location v_i at the time t_i which needs to be split is easily known from the *concretized trace*.

Choosing the variable to split the location When a simulation trajectory differs substantially from the trace obtained by the LP solution, we need to split the location at which the difference is substantial along a hyperplane such that the abstract hybrid automata formed by the linear hybrid automata has a trace that is close to the simulation trajectory. Let D be the metric used to decide if a given location should be refined; then we split those variables into half-spaces which have contributed beyond a threshold to D.

6 Conclusion and Future Work

This early draft discusses the core issues involved in building a CEGAR framework for analyzing nonlinear hybrid systems. The central idea is to use linear programming as a mechanism for obtaining feasible traces of the over-approximate linear hybrid automata (LHA) abstractions and numerical simulation for obtaining a corresponding trace of the original (possibly nonlinear) hybrid system. The distance between these two traces is then used to guide the refinement step in our CEGAR loop.

Several practical issues like the choice of the distance metrics, the choice of picking up a particular solution to the linear program and a characterization of the nonlinear functions which can be handled using this paradigm have been left to a more complete version of this draft. The techniques presented here are also being implemented into a tool which will be a successor to the IRA meta-tool for analyzing LHAs.

7 Acknowledgement

The authors thank Xuandong Li for introducing them to the use of linear programming as an analysis technique for LHAs. Much of the future work benefits from discussions with Ed Clarke and Bruce Krogh.

References

1. R. Alur, C. Courcoubetis, N. Halbwachs, T. A. Henzinger, P.-H. Ho, X. Nicollin, A. Olivero, J. Sifakis, and S. Yovine. The algorithmic analysis of hybrid systems. *Theoretical Computer Science*, 138(1):3–34, 1995.
2. R. Alur, T. Dang, and F. Ivancic. Counter-example guided predicate abstraction of hybrid systems, 2003.

3. Ansgar Fehnker, Edmund M. Clarke, Sumit Kumar Jha, and Bruce H. Krogh. Refining abstractions of hybrid systems using counterexample fragments. In Morari and Thiele [10], pages 242–257.
4. Goran Frehse. Phaver: Algorithmic verification of hybrid systems past hytech. In Morari and Thiele [10], pages 258–273.
5. T.A. Henzinger. The theory of hybrid automata. *Lecture Notes in Computer Science*, page 278, 1996.
6. Thomas A. Henzinger, Pei-Hsin Ho, and Howard Wong-Toi. HYTECH: A model checker for hybrid systems. *International Journal on Software Tools for Technology Transfer*, 1(1–2):110–122, 1997.
7. Pei-Hsin Ho. Automatic analysis of hybrid systems, ph.d. thesis, technical report csd-tr95-1536, cornell university, august 1995, 188 pages, 1995.
8. Sumit Kumar Jha, Bruce Krogh, Jim Weimer, and Edmund M. Clarke. Ira: Iterative relaxation for linear hybrid automata (submitted to hscc 2007).
9. Xuandong Li, Sumit Kumar Jha, and Lei Bu. Towards an efficient path-oriented tool for bounded reachability analysis of linear hybrid systems using linear programming. bmc 2006.
10. Manfred Morari and Lothar Thiele, editors. *Hybrid Systems: Computation and Control, 8th International Workshop, HSCC 2005, Zurich, Switzerland, March 9-11, 2005, Proceedings*, volume 3414 of *Lecture Notes in Computer Science*. Springer, 2005.

Numerical Simulation guided Lazy Abstraction Refinement for Nonlinear Hybrid Automata

Sumit Kumar Jha[1]

Computer Science Department, Carnegie Mellon University
5000 Forbes Avenue, Pittsburgh, PA15213, USA
jha+@cs.cmu.edu

Abstract. This draft suggests a new counterexample guided abstraction refinement (CEGAR) framework that uses the combination of numerical simulation for nonlinear differential equations with linear programming for linear hybrid automata (LHA) to perform reachability analysis on nonlinear hybrid automata. A notion of $\epsilon-$ structural robustness is also introduced which allows the algorithm to validate counterexamples using numerical simulations.

1 Introduction

The model checking of hybrid automata remains a challenge and the existing tools [**?**,**?**] do not scale up to the needs of the industry. Because of the well known fundamental undecidability results [**?**], the model checking of general hybrid automata often proceeds by building successive tighter approximations to these hybrid automata in a relatively easy-to-analyze fragment of hybrid automata like Linear Hybrid Automata [**?**]. Theoretical results about the asymptotic completeness of this approximation procedure form the backbone of such a strategy behind the model checking of nonlinear hybrid automata.

There has been considerable interest in applying Counterexample Guided Abstraction Refinement (CEGAR), which works so well with discrete systems, to the problem of hybrid system verification [**?**]. There has also been some exploration of using fragments instead of counterexamples during abstraction refinement [**?**] and the application of CEGAR specifically to LHA [**?**]. However, our ongoing work makes the following new contributions to the abstraction refinement based analysis of hybrid systems.

– We address the problem of abstraction refinement for nonlinear hybrid automata and use CEGAR to construct successively refined LHA approximations. Our refinement is lazy and hence, refines some parts of the state space more finely than others.
– We use the distance between a feasible path in the abstract linear hybrid automata and the numerically simulated trajectory in the nonlinear hybrid automata to refine those locations in the LHA that do not faithfully represent the behavior of the nonlinear hybrid automata.

– We define a structural notion of robustness and use it to present a counterexample validation algorithm (for a rich class of nonlinear hybrid automata) using linear programming [**?**]. Hence, it is possible to detect reachability of a bad state even before the abstraction refinement loop terminates.

2 Background on LP based path feasibility analysis of LHA

Informally, a linear hybrid automaton is a conventional automaton extended with a set of continuous variables. The states of the automaton called *locations* are annotated with a change rate for each continuous variable such as $\dot{x} = [a, b]$ (x is a variable, and $[a, b]$ is a rational interval), and the transitions of the automaton are labeled with constraints on the variables such as $a \leq \sum_{i=0}^{m} c_i x_i \leq b$ and /or with reset actions such as $x := c$ (x_i and x are variables, a, b, and c_i are real numbers). Such linear hybrid automata are essentially equivalent to the definition given in [**?**]. It is known that this subclass of linear hybrid automata are sufficiently expressive to allow asymptotic completeness of the abstraction process for a general hybrid automata. " A restricted form of linear phase portrait approximations are asymptotically complete, namely, when all automaton constraints are over-approximated using independent, rational lower and upper bounds on the values and derivatives of each variable" [**?**].For simplicity, we suppose that in any linear hybrid automaton considered in this paper, there is just one initial location with no initial conditions and no transitions to the initial location (we assume that each variable with an initial value is reset to the initial value by the transitions from the initial location).

Definition 1. A linear hybrid automaton is a tuple $H = (X, V, E, v_I, \alpha, \beta)$, where

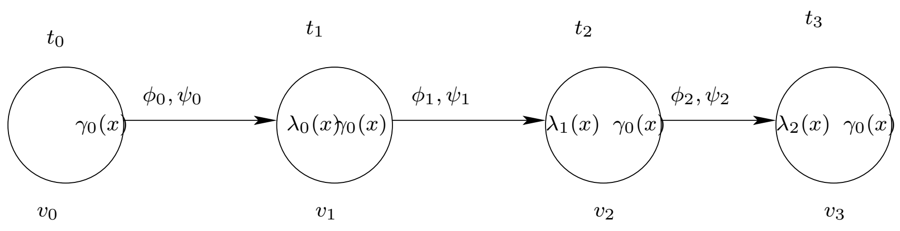
– X is a finite set of real-valued variables.
– V is a finite set of *locations*.
– E is *transition* relation whose elements are of the form (v, ϕ, ψ, v') where v, v' are in V, ϕ is a set of *guards or variable constraints* of the form $a \leq \sum_{i=0}^{m} c_i x_i \leq b$, and ψ is a set of *reset actions* of the form $x := c$ where $x_i \in X$ $(0 \leq i \leq m)$, $x \in X$, a, b and c_i $(0 \leq i \leq m)$ are real numbers, and a and b may be ∞.
– v_I is an *initial* location.
– α is a labeling function which maps each location in $V - \{v_I\}$ to a *state invariant* which is a set of variable constraints of the form $a \leq \sum_{i=0}^{m} c_i x_i \leq b$ where $x_i \in X$ $(0 \leq i \leq m)$, $y \in X$, a, b, and c_i $(0 \leq i \leq m)$ are real numbers, a and b may be ∞.
– β is a labeling function which maps each location in $V - \{v_I\}$ to a set of *change rates* which are of the form $\dot{x}= [a, b]$ where $x \in X$, and a, b are rational numbers $(a \leq b)$. For any location v, for any $x \in X$, there is one and only one change rate definition $\dot{x}= [a, b] \in \beta(v)$.

□

For a linear hybrid automaton $H = (X, V, E, v_I, \alpha, \beta)$, a *path segment* is a sequence of locations

$$v_1 \xrightarrow{(\phi_1,\psi_1)} v_2 \xrightarrow{(\phi_2,\psi_2)} \ldots \xrightarrow{(\phi_{n-1},\psi_{n-1})} v_n$$

which satisfies $(v_i, \phi_i, \psi_i, v_{i+1}) \in E$ for each i $(1 \le i \le n-1)$. A *path* in H is a path segment starting at v_I. The behavior of linear hybrid automata can be represented by *timed sequences*. Any timed sequence is of the form $(v_1, t_1)\hat{}(v_2, t_2)\hat{}\ldots\hat{}(v_n, t_n)$, where v_i $(1 \le i \le n)$ is a location and t_i $(1 \le i \le n)$ is a nonnegative real number, which represents a behavior of an automaton that the system starts at the initial location and changes to the location v_1, stays there for t_1 time units, then changes to the location v_2 and stays in v_2 for t_2 time units, and so on.



Definition 2. *[?]* For a linear hybrid automaton $H = (X, V, E, v_I, \alpha, \beta)$, a timed sequence $(v_1, t_1)\hat{}(v_2, t_2)\hat{}\ldots\hat{}(v_n, t_n)$ represents a behavior of H if the following condition is satisfied:

– there is a path in H of the form

$$v_0 \xrightarrow{(\phi_0,\psi_0)} v_1 \xrightarrow{(\phi_1,\psi_1)} \ldots \xrightarrow{(\phi_{n-1},\psi_{n-1})} v_n \,;$$

– $t_1, t_2, \ldots, t_n$ satisfy all the variable constraints in ϕ_i $(1 \le i \le n-1)$, i.e. for each variable constraint $a \le c_0x_0 + c_1x_1 + \ldots + c_mx_m \le b$ in ϕ_i,

$$\delta_k \le \gamma_i(x_k) \le \delta'_k \text{ for any } k\ (0 \le k \le m), \text{ and}$$
$$a \le c_0\gamma_i(x_0) + c_1\gamma_i(x_1) + \ldots + c_m\gamma_i(x_m) \le b$$

where $\gamma_i(x_k)$ $(0 \le k \le m)$ represents the value of the variable x_k when the automaton stay at v_i with the delay t_i; and, similarly,

– $t_1, t_2, \ldots, t_m, \gamma_i(x_k), \lambda_i(x_k)$ satisfy the state invariant for each location v_i $(1 \le i \le n)$, where $\gamma_i(x_k)$ $(0 \le k \le m)$ represents the value of the variable x_k when the automaton stay at v_i with the delay t_i, and $\lambda_i(x_k)$ $(0 \le k \le m)$ represents the value of the variable x_k after leaving state v_i and after the reset conditions have been applied.

Now, we use linear programming to test the feasibility of a single path for the reachability analysis of linear hybrid automata. Let $H = (X, V, E, v_I, \alpha, \beta)$ be a linear hybrid automaton, , and ρ be a path in H of the form

$$v_0 \xrightarrow{(\phi_0,\psi_0)} v_1 \xrightarrow{(\phi_1,\psi_1)} \dots \xrightarrow{(\phi_{n-1},\psi_{n-1})} v_n$$

where $v_n = v$. For any timed sequence of the form $(v_1, t_1)\hat{}(v_2, t_2)\hat{} \dots \hat{}(v_n, t_n)$, if ρ is feasible, then the following condition must hold:

– $t_1, t_2, \dots, t_n$ satisfy all the variable constraints in $\phi_i (0 \leq i \leq n)$, and
– $t_1, t_2, \dots, t_n$ satisfy all the variable constraints in $\alpha(v_i)$ $(1 \leq i \leq n)$,

which form a group of linear inequalities on $t_1, t_2, \dots, t_n, \gamma_i(x_k), \lambda_i(x_k)$ (see Definition 2), denoted by $\Theta(\rho)$ or $LP_\rho(t_i, \gamma_i(x_k), \lambda_i(x_k))$. It follows that we can check if ρ is a feasible path by checking if the group $\Theta(\rho)$ (or $LP_\rho(t_i, \gamma_i(x_k), \lambda_i(x_k))$) of linear inequalities has a solution, which can be solved by linear programming [**?**].

3 Background on Abstraction of Affine dynamics by LHA

Given a general hybrid system $H = (X, V, E, v_I, \alpha, \beta)$ where X is a finite set of real-valued variables, V is a finite set of *locations*, E is *transition* relation whose elements are of the form (v, ϕ, ψ, v') where v, v' are in V, ϕ is a set of *guards or variable constraints* of the form $a \leq \sum_{i=0}^{m} c_i x_i \leq b$, and ψ is a set of *reset actions* of the form $x := c$ where $x_i \in X$ $(0 \leq i \leq m)$, $x \in X$, a, b and c_i $(0 \leq i \leq m)$ are real numbers, and a and b may be ∞, v_I is an *initial* location, α is a labeling function which maps each location in $V - \{v_I\}$ to a *state invariant* which is a set of variable constraints of the form $a \leq \sum_{i=0}^{m} c_i x_i \leq b$ where $x_i \in X$ $(0 \leq i \leq m)$, $y \in X$, a, b, and c_i $(0 \leq i \leq m)$ are real numbers, a and b may be ∞, β is a labeling function which maps each location in $V - \{v_I\}$ to a set of *change rates* which are of the form $\dot{x_i}= f(x_0, x_1, \dots, x_m, \dot{x}_0, \dot{x}_1, \dots, \dot{x}_m, a_0, a_1 \dots a_m)$ where $x_0, x_1 \dots x_m \in X$, and $a_0, a_1 \dots, a_n$ are real numbers). For any location v, for any $x \in X$, there is one and only one change rate definition.

We construct (in a fashion similar to [**?**,**?**]) a linear hybrid automata H_a which is an over-approximate abstraction of the general hybrid automata H. We define an operator *split* which divides a location into smaller locations and use this to divide each location v_i of the original hybrid automaton.

Definition 3. *Let v be a location and $x_i \in X$ be a variable in a hybrid automata $H = (X, V, E, v_i, \alpha, \beta)$. Suppose $In_v = \{v_0^{in}, v_1^{in} \dots v_n^{in}\}$ be the locations from which there is a transition into the location v and $Out_v = \{v_0^{out}, v_1^{out} \dots v_m^{out}\}$ be the locations to which there is a transition from v. Also, C be a set of linear constraints on X. Then, the operator split (H, v, C) constructs a new hybrid automata $H' = (X, V', E', v_i, \alpha', \beta')$, where*

– $V' = V \cup \{v', v''\} \setminus v$

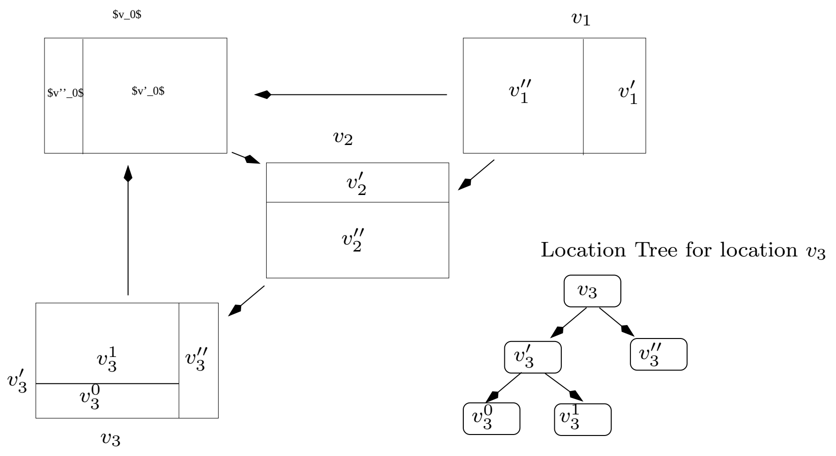
– $\alpha' = \alpha \cup \{\alpha(v'), \alpha(v'')\} \setminus \{\alpha(v)\}$, *where*

- $\alpha(v') = \alpha(v) \cup \{C\}$
- $\alpha(v'') = \alpha(v) \cup \{\neg C\}$

– $E' = E \setminus (\ \{(v_{in}, \phi_{v_{in},v}, \psi_{v_{in},v}, v) | v_{in} \in In_v\} \cup \{(v, \phi_{v,v_{out}}, \psi_{v,v_{out}}, v_{out}) | v_{out} \in Out_v\}\ ) \cup (\ \{(v_{in}, \phi_{v_{in},v}, \psi_{v_{in},v}, v') | v_{in} \in In_v\} \cup \{v', \phi_{v,v_{out}}, \psi_{v,v_{out}}, v_{out} | v_{out} \in Out_v\} \cup \{(v_{in}, \phi_{v_{in},v}, \psi_{v_{in},v}, v'') | v_{in} \in In_v\} \cup \{v'', \phi_{v,v_{out}}, \psi_{v,v_{out}}, v_{out} | v_{out} \in Out_v\}\ ) \cup \{(v', \alpha(v', v'') = C, \{\}, v'')\ , (v'', \alpha(v'', v') = \neg C, \{\}, v')\}$

– $\beta' = \beta \cup \{\beta(v'), \beta(v'')\} \setminus \beta(v)$, *where*

- $\beta(v') = \beta(v)$
- $\beta(v'') = \beta(v)$



We call the new locations v' and v'' as the children of v. In particular, $v' = child(v, C)$ and $v'' = child(v, \neg C)$, where C is the set of linear constraints used to split v. We also call v as the parent of v' and v''. Thus, the *split* operator naturally defines a tree of locations that we call the *location tree* , where the children location are formed by splitting the parent location.

Definition 4. *LHA-approximation to a general hybrid automata: Given a general hybrid automata* $H = (X, V, E, v_i, \alpha, \beta)$, $H_a = (X, V_a, E_a, V_{i_a}, \alpha_a, \beta_a)$ *is a LHA-approximation iff*

– *There exists a hybrid automata* $H' = (X, V', E', V'_i, \alpha', \beta')$, *where* $H' = split^n(H)$.
– $V_a = V', E_a = E', V'_i = V_{i_a}, \alpha' = \alpha_a$
– $\forall v \in V_a, \beta_a(v) \supset \beta'_v$ *and if* $c \in \beta_a(v)$, *then c is of the form* $x := [a, b]$, *where* $a, b \in \mathcal{R}$.

4 Definitions

The linear hybrid automaton H_a is an over-approximate approximation of the general hybrid automata H. A path $\rho = \{v_0, v_1, \ldots v_m\}$, where $v_i \in V$ is said to *exist* in H_a if $(v_i, v_{i+1}) \in E, 0 \leq i < m$.

Consider a path $\rho = \{v_0, v_1, \ldots v_m\}$ that *exists* in the abstract linear hybrid automata model H_a and let $LP_\rho(t_i, \gamma_i(x_k), \lambda_i(x_k))$ be the linear program corresponding to the path. If the linear program LP_ρ has a feasible solution, then the path is said to be *feasible* in the abstract model i.e. the linear hybrid automaton; otherwise it is said to be *infeasible* .

Definition 5. *Trace: Given a feasible path ρ in the abstract linear hybrid automaton model H_a, the feasible solution to the LP program $LP_\rho(t_i, \gamma_i(x_k), \lambda_i(x_k))$ is called a trace of H_a.*
We write trace$(H_a) =< (v_0, \lambda_0(x_0)\ , \lambda_0(x_1) \ldots \lambda_0(x_n), \gamma_0(x_0), \gamma_0(x_1), \ldots \gamma_0(x_n), t_0), (v_1, \lambda_1(x_0), \lambda_1(x_1) \ldots \lambda_1(x_n), \gamma_1(x_0), \gamma_1(x_1), \ldots \gamma_1(x_n), t_1) \ldots \ldots (v_m, \lambda_m(x_0), \lambda_m(x_1) \ldots \lambda_m(x_n), \gamma_m(x_0), \gamma_m(x_1), \ldots \gamma_m(x_n), t_m) >$.

It is known [**?**] that the trace obtained by the linear program is a real execution trace of the over-approximate linear hybrid automata.

Definition 6. *Concretization of a path: Consider a path $\rho = \{v_0, v_1, \ldots v_m\}$ that is feasible in the abstract linear hybrid automata model H_a. Then, the concretization of this path in the original hybrid automata H is the trace $\rho_{concrete} = \{v_0^{root}, v_1^{root} \ldots v_m^{root}\}$, where v_i^{root} is the root of the location tree in which v_i is a leaf.*

As the *split* operator forms a tree of locations in the abstract linear hybrid automata, the root of the location tree is known and the *concretization of an abstract path* is well defined.

Definition 7. *Concretization of a trace: Consider a trace $tr =< (v_0, \lambda_0(x_0)\ , \lambda_0(x_1) \ldots \lambda_0(x_n), \gamma_0(x_0)\ , \gamma_0(x_1), \ldots \gamma_0(x_n), t_0), (v_1, \lambda_1(x_0), \lambda_1(x_1) \ldots \lambda_1(x_n), \gamma_1 (x_0), \gamma_1(x_1), \ldots \gamma_1(x_n), t_1) \ldots \ldots (v_m, \lambda_m(x_0), \lambda_m(x_1) \ldots \lambda_m(x_n), \gamma_m(x_0), \gamma_m(x_1), \ldots \gamma_m(x_n), t_m) >$ corresponding to the path $\rho = \{v_0, v_1, \ldots v_m\}$ that is feasible in the abstract linear hybrid automata model H_a. Then, the concretization of this trace in the original hybrid automata H is the trace $tr_{concrete} =< (v_0^{root}, \lambda_0(x_0), \lambda_0(x_1) \ldots \lambda_0(x_n), \gamma_0(x_0), \gamma_0(x_1), \ldots\ \gamma_0(x_n), t_0), (v_1^{root}, \lambda_1(x_0), \lambda_1(x_1) \ldots \lambda_1(x_n), \gamma_1(x_0), \gamma_1(x_1), \ldots\ \gamma_1(x_n), t_1) \ \ldots \ldots (v_m^{root}, \lambda_m(x_0),\ \lambda_m(x_1) \ \ldots \lambda_m(x_n), \gamma_m(x_0), \gamma_m(x_1), \ldots\ \gamma_m(x_n), t_m) >$, where v_i^{root} is the root of the location tree of which v_i is a leaf.*

Definition 8. *ϵ-Simulation Trajectory: Given a valuation of X i.e. $Val_0(X) = (x_0 = x0, x_1 = x1, \ldots x_n = xn)$ in a location v of a general hybrid system $H = (X, V, E, v_i, \alpha, \beta)$, then $\tau(x0, x1, \ldots xn, v, t)$ is said to be an $\epsilon-$ simulation trajectory for location v with initial valuation $Val_0(X)$ iff*

– $\tau(t = 0) = (x0, x1, \ldots, xn)$

– *if $f(x_0, x_1, \ldots, x_n, t)$ is the solution to the initial value problem $(\beta(v), Val_0(X))$, then $f(t) - \epsilon \leq \tau(t) \leq f(t) + \epsilon$*

It is known that numerical techniques can solve the initial value problem for ODEs (including non-linear ODEs) quiet efficiently.

Definition 9. *$\epsilon-$ Hybrid Simulation Trajectory: Given an initial valuation of X i.e. $Val_0(X) = (x_0 = x0, x_1 = x1, \ldots x_n = xn)$ and a path $\rho = \{v_0, v_1 \ldots v_m\}$ in a general hybrid system $H = (X, V, E, v_i, \alpha, \beta)$, then $\tau(x_0, x_1, \ldots x_n) = f(t)$ is said to be an $\epsilon-$ hybrid simulation trajectory iff*

– *$\tau(0) = Val_0(X)$ and $Val_0(X) \in \alpha(v_0)$*
– *if $x_k := e \in \psi(v_i, v_{i+1})$ then $\tau(x_k, \sum_0^i(t_i)+ = e)$ else $\tau(x_k, \sum_0^i(t_i)+) = \tau(x_k, \sum_0^i(t_i)-)$*
– *Before executing the jump (v_i, v_{i+1}), $\tau(x_k, \sum_0^i(t_i)-)$ satisfies every precondition in $\phi(v_i, v_{i+1})$*
– *Within each location v_i where the timed path has spent time t_i,*
    - *$\forall t, t_i < t < t_{i+1}, \tau(v, \sum_0^i(t_i) + t)$ is an $\epsilon-$ simulation trajectory for location v_i with initial valuation $Val_0 = \tau(\sum_0^i(t_i))$.*

Definition 10. *Guided Simulation Trajectory of the concretization of a trace : An $\epsilon-$ hybrid simulation trajectory τ is said to be a Guided Simulation Trajectory of a concretized trace $tr_{concrete}$ iff*

– *The initial valuation of X i.e. $Val_0(X) = (x_0 = x0, x_1 = x1, \ldots x_n = xn)$ for the trajectory τ is the initial point in the concretized trace $tr_{concrete}$.*
– *The $\epsilon-$ hybrid simulation trajectory τ corresponds to the path $\rho = \{v_0, v_1 \ldots v_m\}$ corresponding to $tr_{cocnrete}$*

5 CEGAR based Refinement of the abstract linear hybrid automata

The CEGAR algorithm repeatedly constructs LHA over-approximations to the given (possibly nonlinear) hybrid system and then asks a LHA analysis engine if the over-approximate LHA admits any counterexample. If it does not, we are done and we report that the original hybrid system has no counterexample either. Otherwise, we take the reported counterexample of the over-approximate LHA and attempt to validate it using numerical simulation. If we succeed in validating the counterexample, we report an error that the bad state is reachable and STOP. Otherwise, we find a location where we need to split the nonlinear hybrid automata and then rebuild a more precise over-approximate abstraction.

```
Algorithm for CEGAR
(Input: Nonlinear Hybrid Automata A. Output: Error  No error
    1. $A_0 = A$; i := 0; L = Universe.
    2. $LHA_i$ = LHA-approximation ($A_i$)
    3. $\mathcal{L}(LHA_i)$ := Language of $LHA_i$. $\mathcal{L}$ represents the set of potential counterexamples in $A_i$ (finitely expressible as a regular expression) [?].
    4. $L = L \cap \mathcal{L}(LHA_i)$
    5. If $L$ is empty, report "BAD STATES NOT REACHABLE" and stop.
    6. Pick a counterexample $ce$ in $L$.
    7. Validate the counterexample $ce$ in the original hybrid automata $A$.
    8. If $ce$ is validated in A, stop and report that ERROR STATE IS REACHABLE.
    9. Compute a refinement operator $split$, and $A_{i+1} = split(A_i)$. Also, compute $L = split(L)$.
    10. i := i + 1
    11. Loop to Step 2.
```

5.1 Counterexample Validation and Structural Robustness

Let $v_0, \gamma_0(x_0), \gamma_0(x_1) \dots \gamma_0(x_n), t_0$ be the initial point in a concretized trace $tr_{concrete}$ for the abstraction i.e. the linear hybrid system H_a corresponding to the general hybrid system H. Let $\tau_{(v_0,x_0,x_1,\dots x_n)}$ be the $\epsilon-$ Hybrid Simulation Trajectory starting from this initial point.

Definition 11. *$\epsilon-$ Structurally Robust Hybrid System: A hybrid system $H = (X, V, E, v_i, \alpha, \beta)$ is said to be structurally robust iff*

- *$\forall i, (v_i, \phi, \psi, v_{i+1}) \in E$, every constraint c in ϕ is satisfied by at least a dense set of size ϵ i.e. If $S = \{Val = (x_0, x_1, \dots x_n) | Val \text{ satisfies } c\}$, then $max_{a \in S} min_{b \in S} d(a, b) > \epsilon$.*
  In particular, we allow only sampled comparisons $x :=_\epsilon c$, which is a shorthand for $\lfloor \frac{c}{\epsilon} \rfloor \times \epsilon < x < \lceil \frac{c}{\epsilon} \rceil \times \epsilon$.

Definition 12. *ϵ Robust Hybrid Simulation Trajectory: Given an initial valuation of X i.e. $Val_0(X) = (x_0 = x0, x_1 = x1, \dots x_n = xn)$ and a path $\rho = \{v_0, v_1 \dots v_m\}$ in a general hybrid system $H = (X, V, E, v_i, \alpha, \beta)$, then $\tau(x_0, x_1, \dots x_n) = f(t)$ is said to be an $\epsilon-$ robust hybrid simulation trajectory iff*

- *$\tau(0) = Val_0(X)$ and $Val_0(X) \in \alpha(v_0)$*
- *if $x_k := e \in \psi(v_i, v_{i+1})$ then $\tau(x_k, \sum_0^i (t_i)+ = e)$ else $\tau(x_k, \sum_0^i (t_i)+) = \tau(x_k, \sum_0^i (t_i)-)$*
- ***Before executing the jump** (v_i, v_{i+1})**,** $\tau(x_k, \sum_0^i (t_i)-)$ **satisfies every precondition in** $\phi(v_i, v_{i+1})$ *$\epsilon-$ robustly .*

  - *A linear constraint c is $\epsilon-$ robustly satisfied by $X = (x_0, x_1, \ldots)$ iff for every X' such that $d(X, X') \leq \epsilon$), $c(X')$ is true.*

– *Within each location v_i where the timed path has spent time t_i,*
  - *$\forall t_i < t < t_{i+1}, \tau(v, \sum_0^i(t_i)+t)$ is an $\epsilon-$ simulation trajectory for location v_i with initial valuation $Val_0 = \tau(\sum_0^i(t_i))$.*

Theorem 1. *If $\tau_{(v_0,x_0,x_1,\ldots x_n)}$ be the $\epsilon-$ Robust Hybrid Simulation Trajectory starting from the initial valuation $Val_0 = \gamma_0(x_0), \gamma_0(x_1) \ldots \gamma_0(x_n)$, and H be a $\epsilon-$ structurally robust hybrid system, then $tr_{concrete}$ corresponds to a* ***real*** *counterexample for the hybrid system H.*

Proof. The proof follows from the definition of $\epsilon-$ robust hybrid automata and the notion of $\epsilon-$ hybrid simulation trajectory.

5.2 Simulation Based Abstraction Refinement

Consider the *concretization of a trace $tr_{concrete}$* with respect to the general hybrid automata H obtained from a *trace tr* of the abstract linear hybrid automata H_a. Also, consider the *guided hybrid simulation trajectory $\tau_{tr_{concrete}}$* corresponding to the concretization of the trace tr with respect to the general hybrid automata H.

Metrics for distance between *trace* and *trajectory* We define two distance metrics between a trace and the corresponding guided hybrid simulation trajectory.

– $D(t) = d(\tau_{tr_{concrete}}(t), tr_{concrete}(t))$.
  This is simply a distance metric between corresponding points on the trace and the trajectory. The metric d may be the Euclidean distance metric or the Manhattan distance metric (linear function).
– $D'(t) = d'(d(\tau_{tr_{concrete}}(t), tr_{concrete}(t)), d(\tau_{tr_{concrete}}(t-), tr_{concrete}(t-))$
  This metric measures how rapidly the guided hybrid simulation trajectory is moving away from the trace. The metric d may be the Euclidean distance metric or the Manhattan distance metric (linear function), while the metric d' may be the real difference. $t-$ represents the last instant of time for which the value of the concretized trace is known.

Strategies to choose the location to be refined Let t_i be the discrete point on the *concretization of a trace* i.e. on $tr_{concrete}$ for which $tr_{concrete}(t)$ is known from the solution of the LP problem. There are few different strategies to choose the location in the approximate linear hybrid automata, where one needs to refine the abstract hybrid automata H_a.

– $min_i|D(t_i)| > \epsilon$, where ϵ is an empirically determined constant.
– $min_i|D'(t_i) - D'(t_{i-1})| > \epsilon$, where ϵ is an empirically determined constant.

– $min_i|D'(t_i)/D'(t_{i-1})| > \epsilon$, where ϵ is an empirically determined constant.

After finding out the point t_i where one needs to refines the location, the location v_i at the time t_i which needs to be split is easily known from the *concretized trace*.

Choosing the variable to split the location When a simulation trajectory differs substantially from the trace obtained by the LP solution, we need to split the location at which the difference is substantial along a hyperplane such that the abstract hybrid automata formed by the linear hybrid automata has a trace that is close to the simulation trajectory. Let D be the metric used to decide if a given location should be refined; then we split those variables into half-spaces which have contributed beyond a threshold to D.

6 Conclusion and Future Work

This early draft discusses the core issues involved in building a CEGAR framework for analyzing nonlinear hybrid systems. The central idea is to use linear programming as a mechanism for obtaining feasible traces of the over-approximate linear hybrid automata (LHA) abstractions and numerical simulation for obtaining a corresponding trace of the original (possibly nonlinear) hybrid system. The distance between these two traces is then used to guide the refinement step in our CEGAR loop.

Several practical issues like the choice of the distance metrics, the choice of picking up a particular solution to the linear program and a characterization of the nonlinear functions which can be handled using this paradigm have been left to a more complete version of this draft. The techniques presented here are also being implemented into a tool which will be a successor to the IRA meta-tool for analyzing LHAs.

7 Acknowledgement

The authors thank Xuandong Li for introducing them to the use of linear programming as an analysis technique for LHAs. Much of the ongoing and future work benefits from discussions with Ed Clarke and Bruce Krogh.